PowerPC™

Freescale Semiconductor, Inc.

Application Note

Avoiding Multiprocessing Paradoxes with the PowerPC 604™ Microprocessor

This application note is intended to be used in conjunction with *Addendum to PowerPC 604 RISC Microprocessor User's Manual: PowerPC 604e™ Microprocessor Supplement and User's Manual Errata* (order number MPC604UMAD/AD). This document describes the following infrequently occurring paradoxes that may occur in a multiprocessing implementation:

- The **lwarx**/**stwcx.** instructions may allow a kill bus operation without modifying the cache block
- An **lwarx** reservation set bus operation may be broadcast without a valid cache entry
- A write-with-kill bus operation may cause a loss of memory coherency

These paradoxes are described briefly in the 604 user's manual addendum. This document provides additional information on how these situations and paradoxes may occur and how they can be avoided.

In this document, the term '604' is used as an abbreviation for 'PowerPC 604 microprocessor'. The PowerPC 604 microprocessors are available from Freescale as MPC604.

To locate any published errata or updates for this document, refer to the website at http://www.mot.com/powerpc/.

© Freescale Semiconductor, Inc., 2004. All rights reserved.

freescale™ semiconductor

The situations and paradoxes discussed in this document follow:

The lwarx/stwcx. Instructions May Allow Kill without Modifying the Cache Block

If a Load Word and Reserve Indexed (**lwarx**) instruction is followed by a Store Word Conditional Indexed (**stwcx.**) instruction to a different address, the 604 can broadcast a kill operation without marking the level 1 (L1) cache block as modified. This may affect some level 2 (L2) cache designs, primarily in multiprocessing systems. This situation is not addressed in the *PowerPC 604 RISC Microprocessor User's Manual*.

Typically, when the 604 issues a kill operation that is not retried ($\overline{\text{ARTRY}}$ asserted) snoop response, it changes the associated L1 cache block from shared (S) to modified (M); for more information, refer to Table 3-2 in the *PowerPC 604 RISC Microprocessor User's Manual*. However, the 604 may issue a kill without modifying the L1 cache block.

This situation occurs when an **lwarx** instruction is followed by an **stwcx.** instruction to a different address. If the **stwcx.** hits on shared (S) data in the data cache while it still has the reservation, the 604 queues a kill operation. However, a snoop operation such as a read-with-intent-to-modify (RWITM) that cancels the reservation may occur without canceling the queued kill. The kill will eventually succeed on the bus but, since the reservation has been lost, the **stwcx.** cannot be performed. Therefore, a kill occurs but the L1 cache block is not marked as modified (M).

Whether system-level problems may occur depends primarily upon the L2 cache design. For example, an L2 cache controller that checks for cache paradoxes could be confused by seeing either multiple kill operations or a kill operation followed by a RWITM to the same L1 cache block.

A less obvious problem can occur when the kill operation causes ownership of a modified cache block to move from an L2 to the 604's L1 without being stored back into memory. This can happen only when an L2 has a modified copy of a cache block and the 604 has a shared copy of the same cache block. If the L2 gives up the cache block when the kill operation occurs, but the 604 fails to mark it as modified, then a snoop or L1 cache block replacement can cause the modified data to be lost. The L2 has given up the cache block and the 604 simply invalidates it since it expects that it has not modified the cache block.

Therefore, to avoid these situations, system designers should not assume that having the 604 issue a kill results in its gaining modified ownership.

An lwarx Reservation Set Bus Operation May Be Broadcast without a Valid Cache Entry

In certain situations, the 604 can broadcast an **lwarx** reservation set (address-only bus operation) although the processor does not have a valid copy of the reservation address in its data cache. This is not described in the *PowerPC 604 RISC Microprocessor User's Manual* which indicates that the processor does not broadcast an **lwarx** reservation set transaction unless the data cache contains a valid copy of the **lwarx** target address.

This condition, as described in the following scenario, can affect systems that check for **lwarx** paradoxes and rely on the processor having a valid copy of the **lwarx** target address when the processor broadcasts an **lwarx** reservation set bus operation.

Scenario:

Assume that the first processor in a two processor system executes an **lwarx** instruction. If the second processor broadcasts a flush (resulting from a **dcbf**) to the same address as the **lwarx** instruction, the first processor may broadcast an **lwarx** reservation set (TT[0–4] = 0b00001) address-only tenure without having a valid copy of the reservation address in its data cache. The flush transaction invalidates the first processor's cache block without canceling the reservation.

For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.

This **lwarx** reservation set broadcast can occur when a snoop-flush to the processor occurs between the time the processor accesses and hits in the data cache for the **lwarx** instruction and the time at which the **lwarx** reservation set transaction gains bus mastership. After a data cache hit for an **lwarx** instruction, the only condition that will cancel the corresponding **lwarx** operation is another snoop clearing the reservation (that is, another processor writes to the reservation address) before the transaction gains mastership of the address bus.

Note that if an in-line L2 cache relies on only the **lwarx** reservation set broadcast to begin snoop filtering for the reservation address, there is a window between the snoop-flush operation to the processor and the **lwarx** reservation set broadcast when the L2 could improperly filter out a snoop to the reservation address on the system bus if it does not look at the state of the reservation ($\overline{\text{RSRV}}$) signal. This is because the reservation address would not be in the L1 and the reservation address is not yet known to the L2. The only way the L2 can know that the processor has a reservation is the $\overline{\text{RSRV}}$ signal.

If the processor detects that a snoop-flush operation to the reservation address has invalidated the cache for the reservation address between the time at which the **lwarx** hit the cache and the time the **lwarx** reservation set is broadcast to the address bus, the processor will always retry the **lwarx** at the cache even though it still performs the **lwarx** reservation set address tenure. In this case, the retried **lwarx** misses in the cache and causes a read-atomic transaction on the bus. Externally this would be seen as the following sequence of operations:

snoop: flush (address A)

processor: **lwarx** reservation set (address A)

processor: read-atomic (address A)

To avoid this situation, **lwarx** paradox checking logic must allow an **lwarx** reservation set operation to be broadcast when the processor can have a valid reservation without a valid copy of the **lwarx** target in its data cache.

A Write-with-Kill Bus Operation May Cause a Loss of Memory Coherency

In some situations, a global write-with-kill operation on the 604 bus can cause a loss of memory coherency. It can appear that a program is not executed serially, which affects the use of global write-with-kill bus operations (direct-memory accesses). Circumstances under which this condition may occur are described as follows:

Scenario 1:

Assume data X is stored at address A and a subsequent store to address A writes data Y into the L1 cache. At this point, the 604 can assert $\overline{\text{ARTRY}}$ and retry a snooped write-with-kill operation to an address in the same cache block as address A and simultaneously invalidate the L1 cache block for address A. If the 604 then tries to load data from address A, it will miss in the L1 cache and the 604 will arbitrate for the bus. If the 604 gains mastership over the write-with-kill operation that was snooped and for which $\overline{\text{ARTRY}}$ was asserted, the load operation will retrieve data X before the data for the write-with-kill is written to memory. Since older data X is returned by the load instead of data Y, it appears that the program is not executed sequentially.

Scenario 2:

When data X is in the 604's copy-back buffer and data Y is in the L1 cache, a write-with-kill operation, in which $\overline{\text{ARTRY}}$ is asserted, causes the data in the copy-back buffer to be pushed to memory and the data in the cache to be killed. The subsequent load retrieves from memory the data that had been in the copy-back buffer.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

The probability of encountering either scenario is increased by performing a **dcbst** to address A before storing data Y. To address this potential situation, software must not attempt to read from a location that may still be in the L1 cache and is the target address for a write-with-kill access (for example, direct-memory access (DMA)). This may be accomplished by flushing the cache block from the cache before the DMA is initiated or by using a software lock to indicate when the DMA is complete and the location is safe for reading.

This occurrence can also be avoided by using write-with-flush instead of write-with-kill bus operations.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.
Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters which may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including “Typicals” must be validated for each customer application by customer’s technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

freescale™
semiconductor